ECS150 Discussion Section

Sophie Engle

( week of 03 November 2003 )
No discussion section

Tuesday, November 11th, 2003
due to Veteran’s Day

(bring questions to Monday’s section)
<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thu 16 Oct 2003</td>
<td>Homework 2 assigned</td>
</tr>
<tr>
<td>Tue 21 Oct 2003</td>
<td>Last chance to resubmit homework 1</td>
</tr>
<tr>
<td>Tue 11 Nov 2003</td>
<td>No discussion section</td>
</tr>
<tr>
<td>Thu 13 Nov 2003</td>
<td><strong>Midterm</strong></td>
</tr>
<tr>
<td>Tue 18 Nov 2003</td>
<td>Homework 2 due</td>
</tr>
<tr>
<td></td>
<td>Homework 3 assigned</td>
</tr>
<tr>
<td>Fri 12 Dec 2003</td>
<td><strong>Final Exam</strong></td>
</tr>
</tbody>
</table>
Upcoming Discussions

- Week of October 20/21 (Sophie)
  - How/where to add system calls

- Week of October 27/28 (Eric)
  - How to modify schedule in kernel

- Week of November 3/4 (Sophie)
  - Miscellaneous homework 2 questions

- Week of November 10 (Eric)
  - Midterm questions
http://wwwcsif.cs.ucdavis.edu/~cs150/
Informal Evaluation

- Would like informal evaluation
  - What works well?
  - What needs to be changed?
  - Comments? Suggestions?

- Turn in sheet of paper
  - Do not include name
  - Please fold in half and turn in at end of class
System call `setLotteryTickets()` must access the process table

- Process table is located inside the kernel
- Must go between user-space to server-space (MM), and then server-space to kernel-space to access the process table

*(see Eric’s slides from last week)*
# Homework 2 Clarification

## Adding MINIX System Call

<table>
<thead>
<tr>
<th>Space</th>
<th>Name/Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER</td>
<td><code>int setLotteryTickets( int _PID, int _tickets ) /* calls do_setLotteryTickets() */</code></td>
</tr>
<tr>
<td>SERVER (MM)</td>
<td><code>int do_setLotteryTickets( void ) /* calls do_syslottery() */</code></td>
</tr>
<tr>
<td>KERNEL</td>
<td><code>int do_syslottery( int _PID, int _tickets ) /* main functionality implemented here */</code></td>
</tr>
</tbody>
</table>
- `setLotteryTickets( pid, tickets )` sets the total number of tickets for the process to `tickets`
- `setLotteryTickets( pid, tickets )` should return the number of tickets actually assigned to process `pid`
Be careful naming parameters...

- **pid** is defined in `/usr/src/mm/param.h`
- **PID** is defined in `/usr/minix/com.h`
Try to name parameters consistent to other system/tasks calls…

- All parameters in /usr/include/unistd.h are prefixed with an underscore (i.e. int _fd)
Recompile \texttt{ps} if you modify \texttt{struct proc} to prevent \texttt{ps} returning garbage

\begin{itemize}
  \item Go to folder \texttt{/usr/src/tools}
  \item Type \texttt{make ps}
  \item Type \texttt{make install}
\end{itemize}
You may be able to avoid having to recompile the libraries for this assignment

Example: Instead of making a library function that makes a taskcall, include `minix/com.h` and use `_taskcall( ... )` directly
Define the constant \texttt{MAX\_TICKETS} so that both the kernel and servers can access it

- Where is \texttt{NR\_PROCS} used?
- Where is \texttt{NR\_PROCS} defined?
Try making a system/task call that returns the total number of tickets currently assigned.

- Useful for debugging
- May be good to implement before attempting `setLotteryTickets(...)`
- May be helpful to determine if `fork()` should fail
Understand how to use `rdy_head[USER_Q]` and the `p_next_ready` attribute of `proc`:

- Useful for lottery scheduling
- Useful for counting total tickets
- Book has entire section on how processes are implemented in Minix
If calculating the total number of tickets using `proc[]`, it may be helpful to:

- Understand `p_priority` in `proc.h`
  - Should only count tickets from valid user processes
- Understand where to start and end searching in the process table
  - See `/usr/src/kernel/main.c` line 06817 for how the kernel accesses `INIT`
Determine if `fork()` should fail before extra memory is allocated for a child process

- See the lecture 2 slides for more on `fork()`
May be able to use `rand()` function in kernel if include `stdlib.h`

- Useful for lottery scheduling algorithm
(look at sample test code)
(test programs will be provided on TA website)
(test program will be provided on TA website)